Journal of Interdisciplinary Mathematics
ISSN: 0972-0502 (Print), ISSN: 2169-012X (Online)
Vol. 27 (2024), No. 6, pp. 1319–1337
DOI : 10.47974/JIM-1782

Accelerating fixed point algorithms for nonlinear Fredholm integral equations

Chinedu Nwaigwe *
Department of Mathematics
Rivers State University
Port Harcourt
Nigeria

Vishnu Narayan Mishra †
Department of Mathematics
Indira Gandhi National Tribal University
Lalpur, Amarkantak, Anuppur 484887
Madhya Pradesh
India

Abstract

This paper proposes a simple idea to speed-up the convergence of a fixed-point iteration for Fredholm equation defined on a mesh. First, the analytical problem is discretized by using quadrature rule and collocating at mesh points. Banach Fixed-Point Theorem is used to construct a discrete Picard scheme (PS). To accelerate the computations, the Picard scheme is modified to form a new faster algorithm christened "Gauss-Seidel-Picard" (GSP) scheme. The GSP scheme modifies the classical Picard iteration in the same way as the Gauss-Seidel iteration modifies the Jacobi iteration. Numerous questions arise, namely: does the GSP scheme converge? If it does converge, is it truly better (faster) than the classical Picard scheme? To address these questions, the convergence errors of the two schemes are derived and it is shown that the GSP scheme converges faster than the Picard scheme. Several numerical experiments are provided to verify this claim in terms of both number of iterations and CPU time. It is found that the GSP solver is much more computationally efficient than the classical scheme. In particular, for the examples considered in the work, the GSP scheme is between 30% to 48% more efficient than the standard Picard scheme. It is suggested that

* *E-mail:* nwaigwe.chinedu@ust.edu.ng (Corresponding Author)
† *E-mail:* vnm@igntu.ac.in

© Taru Publications

this idea be extended to other fixed point methods and to other problems, even in higher dimensions.

Subject Classification: *(2020) 65R20, 45D05.*

Keywords: *Fredholm integral equations, Banach fixed-point theorem, Gauss-seidel-picard scheme, Discrete fixed-point algorithms, Trapezoidal rule, Collocation method, Experimental order of convergence.*

1. Introduction

Consider the following Nonlinear Fredholm integral equation of the second kind:

$$u(x) = g(x) + \int_{y=a}^{b} k(x,y,u(y))dy, \qquad x \in [a,b] \subset \mathbb{R}, \tag{1}$$

where $C[a,b] \ni g : \mathbb{R} \to \mathbb{R}, f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}, k : \mathbb{R} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. Further, we assume that k is Lipschitz continuous with respect to its third argument, namely there exists α_k such that

$$|k(x,y,u) - k(x,y,v)| \leq \alpha_k |u - v| \quad \forall \quad u,v.$$

Integral equations find applications in diverse disciplines, including optimal control and economics [1, 2, 3, 4], epidemiology [5, 6], physical sciences [7, 8, 9], and applied mathematical methods [10, 11]. In general closed form analytical solutions do not exit; numerical techniques provide reliable means to compute an approximated solution [12, 13, 14, 15, 16, 17], see also .

The numerical approximation of nonlinear differential or integral equations usually (naturally) lead to solving nonlinear algebraic systems. Computing the nonlinear systems by Newton-type solvers may involve severe restrictions like differentiability of relevant functions, see [18, 19] for example. Recent studies [11, 20] have shown that fixed point iterations may lead to simple and fast algorithm for approximating nonlinear integral equations. The main advantage of using fixed point methods is that they avoid solving nonlinear systems, even linear system is avoided; every computation is direct and uncoupled. As a result of this uncoupled nature, they are much more easier to implement (program) than the Newton counterparts. The Krasnoselskij iteration has also been constructed for nonlinear functional Volterra equations in [21]; the results are compared with the Newton based proposed in [18]. The results show that the Krasnoselskij fixed point based method are faster and also easier to implement than those of the Newton based method. However both

approaches have comparable accuracy. As the demand for solving large-scale problems would continue to rise, it is also expedient to seek ways of developing faster algorithms. Hence, the crux of this paper is to show how to develop a faster version of a given fixed point method. The Picard iteration is chosen as a case study.

The key idea is to reuse all already computed results in subsequent computations. This is the same idea as the way Gauss-Seidel-Picard linear iterative solver speeds up the Jacobi method. We call the new method Gauss-Seidel-Picard (GSP) scheme. This idea may sound straightforward, but then the question of convergence of the resulting algorithm is an issue that must be addressed both theoretically and numerically, if the scheme is to be accepted. Its numerical advantage over the classical Picard iteration must also be demonstrated in both ways (theoretically and numerically). These are the questions this paper answers. Hence, in section 2, the numerical scheme based on the classical Picard iteration is presented then the scheme is appropriately modified to derive the new GSP scheme. The convergence of the two methods are proved in section 3 where it is shown that the error of classical Picard scheme is an upper bound for the new GSP scheme. In section 4, several numerical examples are provided to numerically demonstrate the convergence of both schemes, and also the faster convergence of the new GSP scheme over the classical Picard method. Performance assessment is also conducted. The concluding remarks are made in section 5.

2. The Numerical Algorithms

The numerical schemes to approximate the solution of problem (1) are presented here. Let $N \in \mathbb{Z}^+$ with $N > 1$ and define the mesh $\Omega_h = \{x_i = a + ih : i = 0,1, \cdots, N, h := \frac{b-a}{N}\}$. Define the grid function

$$\xi_i^N := h\begin{cases}\frac{1}{2}, \text{if } i = 0, N,\\ 1, \text{otherwise,}\end{cases} \text{ for each } i. \tag{2}$$

By the above definition, the following hold:

$$\sum_{j=0}^{N} \xi_j^N = Nh = (b-a), \tag{3}$$

$$\sum_{j=0}^{k} \xi_j^N < (b-a) \text{ provided } k < N. \tag{4}$$

A second order discretization (approximation) of (1) at points in Ω_h is given by (see [22, 12])

$$u(x_i) = g(x_i) + \sum_{j=0}^{N} \xi_j^N k(x_i, x_j, u(x_j)) + O(h^2). \tag{5}$$

We shall refer to (5) as the general nonlinear scheme.

Like (1) which is a nonlinear problem defined on Ω, the approximation (5) is also a nonlinear system but defined on Ω_h. Rather than seek a nonlinear solver which would lead to solving nonlinear systems - a computationally expensive process - we consider iterative techniques that avoid solving algebraic systems. To this end, we first construct the scheme (5) as a fixed point problem and investigate it for contractivity in order to apply Picard-type iterations via the Banach fixed point theorem [23, 24].

Define the operator:

$$Tu(x_i) := g(x_i) + \sum_{j=0}^{N} \xi_j^N k(x_i, x_j, u(x_j)). \tag{6}$$

It is straightforward to use the Lipschitz continuity of k, with respect to u, to show that:

$$\begin{aligned} \| Tu(x_i) - Tv(x_i) \| &= \| \sum_{j=0}^{N} \xi_j^N (k(x_i, x_j, u(x_j)) - k(x_i, x_j, v(x_j))) \| \\ &\leq h\alpha_k \| u - v \| N = (b-a)\alpha_k \| u - v \|. \end{aligned} \tag{7}$$

Hence, by the Banach's fixed point theorem the general scheme (5) has a unique solution that can be approximated by the Picard iteration

$$u_{n+1,i} = g(x_i) + \sum_{j=0}^{N} \xi_j^N k(x_i, x_j, u_{n_j}), \quad n \geq 0, \tag{8}$$

provided

$$(b-a)\alpha_k < 1 \tag{9}$$

holds, where $u_{n,i}$ is an approximation of the point value $u(x_i)$ after n iterations. We shall refer to (8) as the classical Picard scheme.

Obviously, the Picard scheme (8) is easier to implement than the general scheme (5), see Micula [1, 2] in which the Picard algorithm is designed for mixed Volterra-Fredhold equations. Moreover, it has also been shown for nonlinear functional Volterra equations, that both Picard-type schemes like (8) and the general nonlinear scheme (5) have comparable accuracy (see [21]). Hence, the simplicity of fixed point iterations is an

advantage. However, apart from accuracy and simplicity, another desirable feature of a computational algorithm is speed. Hence, it is worth the effort to seek ways of further increasing the rate of convergence of the Picard scheme (8). Therefore, the crux of this study is to leverage on the Picard philosophy to invent an even faster algorithm for efficiently solving problem (1). We propose to adopt the strategy in which the Gauss-Seidel iteration accelerates the Jacobi iteration.

Observe that to compute $u_{n+1,i}$, the Picard scheme (8) uses only the previous iterates $u_{n,j}$ for all $j=0,\cdots,N$. It does not use any solution computed at the new iteration level, $n+1$. Here, we suggest that when computing $u_{n+1,i}$, all the newly computed values before grid point i should be used, that is, to use $u_{n+1,j}$ for all $j = 0,1,2, \cdots, i-1$ since these values are already available. Then for the grid points $j = i,i+1, \cdots, N$ the old values should be used, that is, to use $u_{n,j}$ for $j = i,i+1, \cdots, N$. This is the same idea that the Gauss-Seidel iteration uses to modify the Jacobi iteration to achieve faster convergence. This leads to the following algorithm

$$u_{n+1,i} = g(x_i) + \sum_{j=0}^{i-1} \xi_j^N k(x_i, x_j, u_{n+1,j}) + \sum_{j=i}^{N} \xi_j^N k(x_i, x_j, u_{n,j}) \; \forall i. \tag{10}$$

In this work, we note that

$$\sum_{j=0}^{i-1} \xi_j^N k(x_i, x_j, u_{n,j}) = 0 \text{ whenever } i < 1. \tag{11}$$

We christened this scheme (10) the Gauss-Seidel-Picard scheme (GSPS). The natural questions that arise are (i) on the convergence of both schemes (8) and (10), and (ii) on the advantage of the GSP scheme (10) over the classical Picard's (8). These questions are addressed theoretically in the next section and their numerical verification is given in section 4 with several examples.

3. Convergence Analysis

The current section is devoted to the theoretical proofs of the convergence of schemes (8) and (10), and the advantage of the latter over the former.

3.1 *Preliminaries*

We shall use $u_{n,i}^P$ and $u_{n,i}^G$ to denote the approximation of the point solution $u(x_i)$ after n iterations of the scheme (8) and (10) respectively. Also, denote by

$$e_{n,i}^{P} = |u(x_i) - u_{n,i}^{P}| \text{ and } e_{n,i}^{G} = |u(x_i) - u_{n,i}^{G}| \tag{12}$$

as the error at grid point $x_i \in \Omega_h$ after n iterations of the Picard and GSP schemes respectively. Hence, the maximum Picard and GSP errors after n iterations are given by

$$E_n^{P} = \max_i\{e_{n,i}^{P}\} \text{ and } E_n^{G} = \max_i\{e_{n,i}^{G}\} \tag{13}$$

respectively.

Our primary aim in this section is to derive error estimates of the form

$$e_{n+1,i}^{P} \le C_{n+1,i}^{P} E_n^{P}, \text{ and } e_{n+1,i}^{G} \le C_{n+1,i}^{G} E_n^{G}. \tag{14}$$

for the Picard and GSP schemes respectively. Here, $C_{n+1,i}^{P}$ and $C_{n+1,i}^{G}$ are some constants which would allow us judge both the convergence of the two schemes and also their comparable rates of convergence. For the latter purpose, the following condition shall enable us to judge which of the two schemes is faster.

Definition 3.1: (Comparing Rates of Convergence). We will say that the Picard scheme converges faster than the GSP scheme if $C_{n+1,i}^{G} > C_{n+1,i}^{P}$ at any i. Otherwise, the GSP scheme is faster.

Lemma 3.1: *Suppose the contractivity condition (9) is satisfied, then the following inequality holds*

$$\alpha_k(b-a) - [1-\alpha_k(b-a)] \sum_{j=0}^{i-1} \alpha_k \xi_j^{N} < \alpha_k(b-a) \quad \forall i = 1, \cdots, N. \tag{15}$$

Proof: From the contractivity inequality (9), we deduce that

$$1 - \alpha_k(b-1) < 1. \tag{16}$$

Multiplying inequality (4) by α_k, then multiplying the result by inequality (16) we get the following strict inequality

$$[1-\alpha_k(b-a)] \sum_{j=0}^{i-1} \alpha_k \xi_j^{N} < \alpha_k(b-a). \tag{17}$$

Both sides of the previous inequality are nonzero positive numbers, hence the difference between the larger number $\alpha_k(b-a)$ and the smaller

number $[1-\alpha_k(b-a)]\sum_{j=0}^{i-1}\alpha_k\xi_j^N$ is strictly less than the larger number. That is,

$$\left[\alpha_k(b-a)-[1-\alpha_k(b-a)]\sum_{j=0}^{i-1}\alpha_k\xi_j^N\right]<\alpha_k(b-a)\;\forall i=1,\cdots,N.$$

as claimed.

3.2 *Analysis of the Picard Scheme*

Lemma 3.2: *Under the contractivity assumption (9), the Picard error satisfies*

$$e_{n+1,i}^P\le C_{n+1,i}^P E_n^P+O(h^2) \tag{18}$$

where the coefficient is given by

$$C_{n+1,i}^G=\alpha_k(b-a)\quad\text{for all } i=0,1,2,\cdots,N. \tag{19}$$

Proof: By definition

$$\begin{aligned}
e_{n+1,i}^P &:= |u(x_i)-u_{n+1,i}^P| = |\sum_{j=0}^N \xi_j^N(k(x_i,x_j,u(x_j))-k(x_i,x_j,u_{n,j}^P))+O(h^2)|\\
&\le \sum_{j=0}^N \xi_j^N\alpha_k|u(x_j))-u_{n,j}^P|+O(h^2)=\sum_{j=0}^N\xi_j^N\alpha_k e_{n,j}^P+O(h^2)\\
&\le E_n^P\sum_{j=0}^N\xi_j^N\alpha_k+O(h^2)\\
&=\alpha_k(b-a)E_n^P+O(h^2)\\
&=C_{n+1,i}^P E_n^P+O(h^2),
\end{aligned} \tag{20}$$

where

$$C_{n+1,i}^P=\alpha_k(b-a)\qquad\text{for all } i=0,1,\cdots,N.$$

Since $C_{n+1,i}^P=\alpha_k(b-a)<1$, inequality (20) shows that the new approximation error $e_{n+1,i}^P$ at each grid point, x_i, is less than the maximum error in the previous errors iteration. This proves that if the contractivity condition (9) holds, then the Picard scheme (8) converges to the exact solution. This is in perfect agreement with the statement of the Banach fixed point theorem at the discrete level.

3.3 *Analysis of the Gauss-Seidel-Picard Scheme*

Lemma 3.3: *Under the contractivity assumption (9), the GSP error satisfies*

$$e^G_{n+1,i} \le C^G_{n+1,i} E^G_n + O(h^2) \tag{21}$$

where the coefficient satisfies the strict inequality

$$C^G_{n+1,i} < \alpha_k (b-a) \quad \text{for all } i = 1,2,\cdots,N \tag{22}$$

except at $i = 0$ *where equality holds, namely*

$$C^G_{n+1,0} = \alpha_k (b-a). \tag{23}$$

Proof: By definition

$$\begin{aligned} e^G_{n+1,i} &:= |\, u(x_i) - u^G_{n+1,i} \,| = |\sum_{j=0}^{i-1} \xi^N_j (k(x_i, x_j, u(x_j)) - k(x_i, x_j, u^G_{n+1,j})) \\ &+ \sum_{j=i}^{N} \xi^N_j (k(x_i, x_j, u(x_j)) - k(x_i, x_j, u^G_{n,j})) + O(h^2)\,| \\ &\le \alpha_k \sum_{j=0}^{i-1} \xi^N_j \,|\, u(x_j) - u^G_{n+1,j} \,| + \alpha_k \sum_{j=i}^{N} \xi^N_j \,|\, u(x_j) - u^G_{n+1,j} \,| + O(h^2), \\ &\le \alpha_k \sum_{j=0}^{i-1} \xi^N_j e^G_{n+1,j} + \alpha_k \sum_{j=i}^{N} \xi^N_j e^G_{n,j} + O(h^2), \forall i = 0,\cdots,N. \end{aligned} \tag{24}$$

To understand the behavior of the GSP error, $e^G_{n+1,i}$, we take the error equation (24) in turns for $i = i = 0, 1, \cdots, N$. Setting $i = 0$ in (24), one gets

$$\begin{aligned} e^G_{n+1,0} &\le \alpha_k \sum_{j=0}^{-1} \xi^N_j e^G_{n+1,j} + \alpha_k \sum_{j=0}^{N} \xi^N_j e^G_{n,j} + O(h^2), \\ &= \alpha_k \sum_{j=0}^{N} \xi^N_j e^G_{n,j} + O(h^2), \qquad (by\ (11)) \\ &\le \alpha_k \sum_{j=0}^{N} \xi^N_j E^G_n + O(h^2) = \alpha_k E^G_n \sum_{j=0}^{N} \xi^N_j + O(h^2) \\ &\le \alpha_k (b-a) E^G_n + O(h^2) \qquad (25) \\ &= C^G_{n+1,0} E^G_n + O(h^2). \qquad (26) \end{aligned}$$

with

$$C^G_{n+1,0} = \alpha_k (b-a). \tag{27}$$

This proves (23).

Setting $i = 1$ in (24), one gets

$$\begin{aligned} e^G_{n+1,1} &\le \alpha_k \xi^N_0 e^G_{n+1,0} + \alpha_k \sum_{j=1}^{N} \xi^N_j e^G_{n,j} + O(h^2) \\ &\le \xi^N_0 \alpha_k^2 (b-a) E^G_n + (\alpha_k (b-a) - \xi^N_0 \alpha_k) E^G_n + O(h^2) \\ &= [\alpha_k (a-b) - (1 - \alpha_k (a-b)) \xi^N_0 \alpha_k] E^G_n + O(h^2) \\ &= C^G_{n+1,1} E^G_n + O(h^2), \end{aligned} \tag{28}$$

where

$$C^G_{n+1,1} := [\alpha_k (a-b) - (1 - \alpha_k (a-b)) \xi^N_0 \alpha_k]. \tag{29}$$

Setting $i = 1$ in Lemma 3.1, one gets

$$C^G_{n+1,1} < \alpha_k (b-a). \tag{30}$$

Similarly,

$$\begin{aligned} e^G_{n+1,2} &\le \alpha_k \sum_{j=0}^{1} \xi^N_j e^G_{n+1,j} + \alpha_k \sum_{j=2}^{N} \xi^N_j e^G_{n,j} + O(h^2) \\ &\le \xi^N_0 \alpha_k^2 (b-a) E^G_n + \xi^N_1 \alpha_k C^G_{n+1,1} E^G_n + \alpha_k (b-a) E^G_n - (\xi^N_0 + \xi^N_1) \alpha_k E^G_n + O(h^2) \\ &\le [\alpha_k (b-a) + \alpha_k (b-a) \xi^N_0 \alpha_k + C^G_{n+1,1} \xi^N_1 \alpha_k - \alpha_k (\xi^N_0 + \xi^N_1)] E^G_n + O(h^2) \\ &< [\alpha_k (b-a) + \alpha_k (b-a) [\xi^N_0 + \xi^N_1] \alpha_k - [\xi^N_0 + \xi^N_1] \alpha_k] E^G_n + O(h^2) \\ &= [\alpha_k (b-a) - [1 - \alpha_k (b-a)] \sum_{j=0}^{1} \xi^N_j \alpha_k] E^G_n + O(h^2) \\ &= C^G_{n+1,2} E^G_n + O(h^2), \end{aligned} \tag{31}$$

where

$$C^G_{n+1,2} := [\alpha_k (b-a) - (1 - \alpha_k (b-a)) \sum_{j=0}^{1} \xi^N_j \alpha_k]. \tag{32}$$

Again, setting $i = 2$ in Lemma 3.1, one gets

$$C^G_{n+1,2} < \alpha_k (b-a). \tag{33}$$

We have proved this Lemma for $i = 1, 2,$ namely

$$C^G_{n+1,j} < \alpha_k (b-a), \qquad i = 1, 2. \tag{34}$$

We now use the above result to prove the Lemma for $i \ge 3$ as follows:

$$e^G_{n+1,i} \le \alpha_k \sum_{j=0}^{i-1} \xi^N_j e^G_{n+1,j} + \alpha_k \sum_{j=i}^{N} \xi^N_j e^G_{n,j} + O(h^2)$$

$$
\begin{aligned}
&\leq \alpha_k \sum_{j=0}^{i-1} \xi_j^N C_{n+1,j}^G E_n^G + \alpha_k \sum_{j=i}^{N} \xi_j^N E_n^G + O(h^2) \\
&= \alpha_k \sum_{j=0}^{i-1} \xi_j^N C_{n+1,j}^G E_n^G + \alpha_k \sum_{j=0}^{N} \xi_j^N E_n^G - \alpha_k \sum_{j=0}^{i-1} \xi_j^N E_n^G + O(h^2) \\
&< \left[\alpha_k (b-a) + \alpha_k \sum_{j=0}^{i-1} \xi_j^N (C_{n+1,j}^G - 1) \right] E_n^G + O(h^2) \\
&< \left[\alpha_k (b-a) + \alpha_k \sum_{j=0}^{i-1} \xi_j^N [\alpha_k (b-a) - 1] \right] E_n^G + O(h^2) \\
&= \left[\alpha_k (b-a) - [1 - \alpha_k (b-a)] \sum_{j=0}^{i-1} \alpha_k \xi_j^N \right] E_n^G + O(h^2) \\
&= C_{n+1,i}^G E_n^G + O(h^2), \qquad (35)
\end{aligned}
$$

$$
C_{n+1,i}^G := \left[\alpha_k (b-a) - [1 - \alpha_k (b-a)] \sum_{j=0}^{i-1} \alpha_k \xi_j^N \right]. \qquad (36)
$$

By Lemma 3.1, we get that $C_{n+1,i}^G < \alpha_k (b-1)$ for $i \geq 2$. Hence, the result is proved for all $i = 0, 1, 2, \cdots, N$.

The above analyses lead to the following theorem.

Theorem 3.1: (Main Result). *Suppose the contractivity condition (9) holds, then*

(i) *both the Picard scheme (8) and the Guass-Seidel-Picard (GSP) scheme (10) converge to the exact solution of problem (1).*

(ii) *the GSP scheme converges faster than the Picard scheme.*

$$
C_{n+1,i}^G < C_{n+1,i}^P \text{ for all } i = 1, 2, \cdots, N \qquad (37)
$$

(see definition (3.1)).

Proof:

(i) From Lemmas 3.2 and 3.3, the error of both schemes satisfy

$$
\begin{aligned}
e_{n+1,i}^D &\leq C_{n+1,i}^D E_n^D + O(h^2), \\
&\leq \alpha_k (b-a) E_n^D + O(h^2) \quad \text{for all } i \text{ and } D = P, G.
\end{aligned}
$$

In particular,

$$
E_{n+1}^D = \max_i (e_{n+1,i}^D) \leq \alpha_k (b-a) E_n^D + O(h^2) \quad \text{for all } i \text{ and } D = P, G.
$$

By discrete Grownwall inequality, we get

$$E_n^D \le [\alpha_k (b-a)]^n E_0^D + O(h^2), \quad \text{for } D = P, G.$$

By the contractivity assumption (9), we have

$$E_n^D \to 0 \text{ as } n \to \infty \text{ and } h \to 0 \text{ for } D = P, G.$$

Hence, both schemes converge.

(ii) From Lemma 3.2 the error coefficient of the Picard scheme is

$$C_{n+1,i}^P = \alpha_k (b-a),$$

while that of the GSP scheme is

$$C_{n+1,0}^P = \alpha_k (b-a), \text{ and } C_{n+1,i}^P < \alpha_k (b-a) \text{ for } i > 0,$$

(see Lemma 3.3).

Therefore,

$$C_{n+1,i}^G < C_{n+1,i}^P, \text{for all } i > 0,$$

equality only holds at $i = 0$. Hence, by definition 3.1, the GSP scheme (10) converges faster than the Picard scheme (8).

4. Numerical Experiments

This section presents several numerical experiments to verify the theoretical results obtained in the previous section. For each example, a sequence of solutions is computed with decreasing mesh sizes. The fixed point iteration is taken to converge and terminated whenever the 2-norm, $\| \vec{u}_{n+1} - \vec{u}_n \|_2$, is less than 8×10^{-11}, that is whenever

$$\sqrt{\sum_{i=0}^{N} | u_{n+1,i} - u_{n,i} |^2} \le 8 \times 10^{-11}.$$

The error is computed in maximum norm, $\max_i | u(x_i) - u_{n,i} |$, where $u_{n,i}$ is the converged solution at grid point x_i. Finally, the experimental order of convergence is computed as (see [25, 19])

$$EOC = \frac{log\left(\frac{e_h}{e_{h/2}}\right)}{log(2)}. \tag{38}$$

The examples are constructed via the method of manufactured solutions [26, 27, 28, 29, 30].

4.0.1 Example 1

The first example is the following:

$$u(x) = g(x) + \int_0^1 u(y) e^{x-y-u^2} dy, \quad x \in [0,1]. \tag{39}$$

where

$$g(x) = -e^{x-e^{-2}} / 2 + e^{x-1} / 2 + e^{-x}.$$

The exact solution of this problem is $u(x) = e^{-x}$. Table 1 tabulates the numerical results and their errors and EOC. The results show that both the Picard and GSP schemes converge but the GSP scheme converges after a fewer number of iterations than the Picard scheme. This verifies the theorems of section 3. Further, the results also show that both methods have comparable accuracy, hence the determining factor of superiority is the speed.

Table 1

Numerical Results for Example 1. N = number of sub-intervals.

N	GPS Error	GPS EOC	GPS Iters	Picard Error	Picard EOC	Picard Iters
2	0.0127958	-	12	0.0127958	-	15
4	0.0033999	1.912123	12	0.0033999	1.912123	13
8	0.0008599	1.983257	11	0.0008599	1.983257	13
16	0.0002156	1.996103	11	0.0002156	1.996103	14
32	5.3923847e-05	1.999043	11	5.3923846e-05	1.999043	14
64	1.3483187e-05	1.999762	10	1.3483185e-05	1.999762	14
128	3.3709357e-06	1.999941	10	3.3709338e-06	1.999941	14
256	8.4274261e-07	1.999985	10	8.4274230e-07	1.999985	15
512	2.1068620e-07	1.999996	11	2.1068588e-07	1.999998	15
1024	5.2671584e-08	1.999999	11	5.2671262e-08	2.000006	15
2048	1.3167896e-08	2.000000	11	1.3167579e-08	2.000026	15

4.0.2 Example 2

In this example, we consider the problem:

$$u(x) = \left(\frac{6-\log(2)}{6}\right)x^3 + \int_0^1 \left(\frac{x^3 y^5}{1+u^2(y)}\right) dy, \quad x \in [0,1]. \tag{40}$$

The exact solution of this problem is $u(x) = x^3$. The numerical results which are shown in Table 2 also verify that both methods converge with second order of accuracy, but the GSP scheme converges faster than the Picard's.

Table 2

Numerical Results for Example 2

N	GPS Error	GPS EOC	GPS Iters	Picard Error	Picard EOC	Picard Iters
2	0.022116617	-	13	0.022115617	-	13
4	0.004893735	2.1760577	10	0.004893735	2.1760577	11
8	0.001223849	1.9995099	9	0.001223849	1.9995099	10
16	0.000305866	2.0004536	8	0.000305866	2.0004536	10
32	7.645992774e-05	2.0001248	8	7.645992674e-05	2.000125	10
64	1.911456095e-05	2.0000318	8	1.911455998e-05	2.000032	10
128	4.778613820e-06	2.0000080	7	4.778612848e-06	2.000008	10
256	1.194651808e-06	2.0000020	7	1.194651862e-06	2.000002	11
512	2.986628536e-07	2.0000005	7	2.986629086e-07	2.000000	11
024	7.466570495e-08	2.0000002	8	7.466576713e-08	1.999999	11
2048	1.866642574e-08	2.0000000	8	1.866648791e-08	1.999996	11
4096	4.666606435e-09	2.0000000	8	4.666668607e-09	1.999986	11

4.0.3 Example 3

As a third example, we consider the problem:

$$u(x) = g(x) + \int_0^1 k(x, y, u(y))dy, \quad x \in [0,1], \tag{41}$$

where

$$k(x, y, u(y)) = x^4 + y^2 - \frac{1}{3}u^3(y) + u^2(y)$$

and

$$g(x) = -x^4 + \cos(x) - \frac{5}{6} - \frac{\sin(2)}{4} + \frac{\sin(3)}{36} + \frac{\sin(1)}{4}.$$

The exact solution of this problem is $u(x) = \cos(x)$ and the results are displayed in Table 3. One can also see that the GSP scheme is almost twice

as fast as the Picard scheme for this particular problem. Indeed, the GSP scheme is a provable improvement on the Picard algorithm.

Table 3

Numerical Results for Example 3.

N	GPS Error	GPS EOC	GPS Iters	Picard Error	Picard EOC	Picard Iters
2	0.4982756	-	104	0.4982756	-	151
4	0.2436639	1.03205	436	0.2436639	1.032	696
8	0.0446778	2.44726	327	0.0446778	2.447	565
16	0.0100999	2.14522	255	0.0100999	2.145	462
32	0.0024684	2.03268	241	0.0024684	2.033	448
64	0.0006137	2.00796	239	0.0006137	2.008	450
128	0.0001532	2.00198	240	0.0001532	2.002	457
256	3.8291066e-05	2.00050	243	3.8291009e-05	2.001	464
512	9.5719249e-06	2.00013	246	9.5718840e-06	2.000	471
1024	2.3929149e-06	2.00004	250	2.3928860e-06	2.000	479
2048	5.9821497e-07	2.00003	254	5.9819480e-07	2.000	487

4.1 *Performance Analysis*

In this subsection the two methods are compared in terms of CPU time. For each problem a mesh is chosen and the CPU time for computing the solution using each method is recorded. This is repeated over a number of times and the average CPU time is computed for each method on the chosen mesh. The efficiency of the GSP scheme over the Picard scheme is calculated as

$$\%\ \text{increase by } GSP = \frac{T^P - T^G}{T^P} \times 100\%,$$

where T^P and T^G are the calculated average CPU time for the Picard and GSP schemes respectively. This is repeated for different meshes. Finally, the average efficiency over all the meshes is computed. The results for the three examples above are displayed in Tables 4-6.

The results show that for example 1, the GSP has an average efficiency of about 30% over the Picard scheme (see Table 4). The average efficiency of GSP over Picard's for examples 2 and 3 are 33.5% and 47.9%, see Tables 5 and 6 respectively. These support the theoretical results that the GSP scheme (10) is a faster algorithm than the Picard scheme (8).

Table 4

CPU times for Example 1: averaged over ten computations.

N	GPS CPU Time	Picard CPU Time	% increase by GSP	GSP Error	Picard Error
100	0.23665032386779786	0.3298926591873169	28.264446850445058	5.64e-06	5.64e-06
250	1.4610035419464111	2.1701683044433593	32.67786931755261	8.91e-07	8.91e-07
350	2.846740961074829	4.2437649965286255	32.91944856976185	4.54e-07	4.54e-07
400	3.7062979459762575	5.535623574256897	33.04642383538893	3.47e-07	3.47e-07
500	6.352642893791199	8.637668561935424	26.45419480684757	2.22e-07	2.22e-07
1000	25.34214859008789	34.464588689804074	26.469023558708383	5.53e-08	5.53e-08
	Average Efficiency is		29.971901156450738		

Table 5

CPU times for Example 2: averaged over ten computations.

N	GPS CPU Time	Picard CPU Time	% increase by GSP	GSP Error	Picard Error
100	0.09763574600219727	0.13880887031555175	29.661738633674023	7.99e-06	7.99e-06
250	0.5996782064437867	0.9408061742782593	36.25911236139255	1.26e-06	1.26e-06
350	1.187613558769226	1.8578848361968994	36.07711653429081	6.43e-07	6.43e-07
400	1.5536530256271361	2.429626226425171	36.053825533769334	4.92e-07	4.92e-07
500	2.4257277488708495	3.799241065979004	36.15230761236841	3.14e-07	3.14e-07
1000	11.094605803489685	15.19397246837616	26.980216486627544	7.85e-08	7.85e-08
	Average Efficiency is		33.530719527020445		

Table 6

CPU times for Example 3: averaged over ten computations.

N	GPS CPU Time	Picard CPU Time	% increase by GSP	GSP Error	Picard Error
100	4.683212924003601	8.970112681388855	47.79092425761487	0.000256	0.000256
250	29.18707368373871	56.01472322940826	47.89392502359948	4.05e-05	4.05e-05
350	57.49981915950775	110.61737747192383	48.01918064446793	2.06e-05	2.06e-05
400	75.38268032073975	145.01007242202758	48.015555704743704	1.58e-05	1.58e-05
500	118.70616343021393	227.65521986484526	47.857042987774406	1.01e-05	1.01e-05
	Average Efficiency is		47.91532572364007		

5. Conclusion

Two fixed point based algorithms have been proposed for solving nonlinear Fredholm integral equations at grid points. Both schemes are analyzed in detail and shown to converge, with the nonstandard scheme converging faster than the standard Picard iteration. The novelties of this work include (i) proposing the Picard scheme (8) for problem (1), (ii) the design of a strategy to accelerate the Picard scheme on a one dimensional grid leading to the nonstandard scheme (10), and (iii) their detailed theoretical analysis and numerical verification. This opens up a new line of thinking in the use of fixed point methods for discrete problems. Numerical experiments show that indeed

(1) both schemes converge to the exact solution, but
(2) the newly proposed nonstandard (GSP) scheme converges faster than the classical scheme, and
(3) for the problems considered here, the GSP scheme is between 30% to 48% more efficient than the standard Picard scheme.

Since both schemes have comparable accuracy while the GSP is considerably faster, we suggest that the GSP scheme should always be preferred over the Picard scheme whenever grid based problems are considered. We recommend that this approach be extended to other iterative algorithms. Finally, since this idea is borrowed from the techniques in iterative linear solvers, we also recommend that more research should be geared towards extending ideas from more sophisticated iterative solvers to accelerating fixed point methods.

Conflict of Interest

There is no conflict of interest.

References

[1] Shouchuan Hu, Mohammad Khavanin, and WAN Zhuang. Integral equations arising in the kinetic theory of gases. *Applicable Analysis*, 34(3-4) : 261-266 (1989).

[2] Abdul Jerri. Introduction to integral equations with applications. John Wiley & Sons (1999).

[3] Donal Oregan. Existence results for nonlinear integral equations. *Journal of Mathematical Analysis and Applications*, 192(3) : 705-726 (1995).

[4] Jan Prüss. Evolutionary integral equations and applications, volume 87. Birkhäuser (2013).

[5] K Maleknejad and M Hadizadeh. A new computational method for volterra-fredholm integral equations. *Computers & Mathematics with Applications*, 37(9) : 1-8 (1999).

[6] Abdul-Majid Wazwaz. A reliable treatment for mixed volterra–fredholm integral equations. *Applied Mathematics and Computation*, 127(2-3) : 405-414 (2002).

[7] MA Abdou. On a symptotic methods for fredholm{volterra integral equation of the second kind in contact problems. *Journal of Computational and Applied Mathematics*, 154(2) : 431-446 (2003).

[8] Tien-Dung Le, Christian Moyne, MA Murad, and SA Lima. A twoscale non-local model of swelling porous media incorporating ion size correlation effects. *Journal of the Mechanics and Physics of Solids*, 61(12) : 2493-2521 (2013).

[9] Aline C Rocha, Marcio A Murad, Christian Moyne, Saulo P Oliveira, and Tien D Le. A new methodology for computing ionic profiles and disjoining pressure in swelling porous media. *Computational Geosciences*, 20(5) : 975-996 (2016).

[10] Abdul-Majid Wazwaz. First Course In Integral Equations, A. World Scientific Publishing Company (2015).

[11] Sanda Micula. On some iterative numerical methods for mixed volterra–fredholm integral equations. *Symmetry*, 11(10) : 1200 (2019).

[12] Chinedu Nwaigwe and Deborah Ngochinma Benedict. Generalized banach fixed-point theorem and numerical discretization for nonlinear volterra-fredholm equations. *Journal of Computational and Applied Mathematics*, page 115019 (2022).

[13] Kazemi Manochehr. Triangular functions for numerical solution of the nonlinear volterra integral equations. *Journal of Applied Mathematics and Computing*, pages 1-24 (2021).

[14] Yadollah Ordokhani and Mohsen Razzaghi. Solution of nonlinear volterra-fredholm-hammerstein integral equations via a collocation method and rationalized haar functions. A*pplied Mathematics Letters*, 21(1) : 4-9 (2008).

[15] Hermann Brunner. On the numerical solution of nonlinear volterra–fredholm integral equations by collocation methods. *SIAM Journal on Numerical Analysis*, 27(4) : 987-1000 (1990).

[16] R Ezzati and S Najafalizadeh. Numerical methods for solving linear and nonlinear volterra-fredholm integral equations by using cas wavelets. *World Applied Sciences Journal*, 18(12) : 1847-1854 (2012).

[17] SC Shiralashetti and Lata Lamani. Cas wavelets stochastic operational matrix of integration and its application for solving stochastic itô-volterra integral equations. *Jordan Journal of Mathematics and Statistics* (JJMS), 14(3) : 555-580 (2021).

[18] Sohrab Bazm, Pedro Lima, and Somayeh Nemati. Analysis of the euler and trapezoidal discretization methods for the numerical solution of nonlinear functional volterra integral equations of urysohn type. *Journal of Computational and Applied Mathematics*, page 113628 (2021).

[19] Chinedu Nwaigwe. Solvability and approximation of nonlinear functional mixed volterra-fredholm equation in banach space. *Journal of Integral Equations and Applications*, 34(4) : 489-500 (2022).

[20] Sanda Micula. Numerical solution of two-dimensional fredholm-volterra integral equations of the second kind. *Symmetry*, 13(8) : 1326 (2021).

[21] Chinedu Nwaigwe and Sanda Micula. Fast and accurate numerical algorithm with performance assessment for nonlinear functional volterra equations. https://www.researchgate.net/publication/361722935_Fast_and_Accurate_Numerical_Algorithm_with_Performance_Assessment_for_Nonlinear_Functional_Volterra_Equations, 2023. (Accessed on 16-February-2023).

[22] Richard L Burden. Numerical analysis (2011).

[23] Vasile Berinde. Iterative Approximation of Fixed Points. Springer (2007).

[24] Godwin Amechi Okeke. Convergence analysis of the picard-ishikawa hybrid iterative process with applications. *Afrika Matematika*, 30(5) : 817-835 (2019).

[25] Chinedu Nwaigwe. Coupling methods for 2d/1d shallow water flow models for flood simulations. PhD thesis, University of Warwick (2016).

[26] Patrick J Roache. The method of manufactured solutions for code verification. In Computer Simulation Validation, pages 295-318. Springer (2019).

[27] Chinedu Nwaigwe and Azubuike Weli. Analysis of two finite difference schemes for a channel flow problem. *Asian Research Journal of Mathematics*, 15:1-14 (2019).

[28] C Nwaigwe, RI Ndu, and A Weli. Wall motion effects on channel flow with temperature-dependent transport properties. *Applied Mathematics*, 9(3) : 162-168 (2019).

[29] Kambiz Salari and Patrick Knupp. Code verification by the method of manufactured solutions. Technical report, Sandia National Labs., Albuquerque, NM (US); Sandia National Labs (2000).

[30] Chinedu Nwaigwe. An unconditionally stable scheme for two-dimensional convection-diffusion-reaction equations. https://www.researchgate.net/publication/357606287_An_Unconditionally_Stable_Scheme_for_Two-Dimensional_Convection-Diffusion-Reaction_Equations, 2022. (Accessed on 19-July-2022).

Received May, 2023